**CƠ SỞ TRÍ TUỆ NHÂN TẠO (CQ2017/21)**

BÁO CÁO ĐỒ ÁN

ỨNG DỤNG CÁC GIẢI THUẬT TÌM KIẾM

ROBOT TÌM ĐƯỜNG

**GVHD:** Lê Ngọc Thành

C:\Users\tdqua_000\Dropbox\SS-Slides\DeCuong-CDIO\Template CDIO v4.2\Templates\Hinh anh\LogoTruong.png

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

**BÁO CÁO ĐỒ ÁN 1**

# Thông tin nhóm

|  |  |  |
| --- | --- | --- |
| **STT** | **MSSV** | **Họ và tên** |
| 1 | 1712529 | Nguyễn Trọng Khiêm |
| 2 | 1712513 | Nguyễn Duy Khải |
| 3 | 1712466 | Nguyễn Hữu Huân |

# Mức độ hoàn thành các yêu cầu

|  |  |  |
| --- | --- | --- |
| **Mức** | **Yêu cầu** | **Mức độ hoàn thành** |
| 1 | cài đặt thành công 1 thuật toán để tìm đường đi từ S tới G. | 100% |
| 2 | cài đặt ít nhất 3 thuật toán khác nhau (ví dụ tìm kiếm mù, tham  lam, heuristic, …) | 100% |
| 3 | trên bản đồ sẽ xuất hiện thêm một số điểm khác được gọi là  điểm đón. Xuất phát từ S, sau đó đi đón tất cả các điểm này rồi đến trạng thái  G. Thứ tự các điểm đón không quan trọng. Mục tiêu là tìm ra cách để tổng  đường đi là nhỏ nhất | 100% |
| 4 | các hình đa giác có thể di động được với tốc độ h  tọa độ/s. Cách thức di động có thể ở mức đơn giản nhất là tới lui một khoảng  nhỏ để đảm bảo không đè lên đa giác khác. Chạy ít nhất 1 thuật toán trên đó. | 0% |
| 5 | thể hiện mô hình trên không gian 3 chiều (3D). | 0% |

# Chi tiết thực hiện

## 3.1 Breadth First Search (BFS):

1. Thuật toán:

Quy ước để dễ dàng trong trình bày:

* Open: là tập hợp các đỉnh chờ được xét ở bước tiếp theo theo hàng đợi (hàng đợi: dãy các phần tử mà khi thêm phần tử vào sẽ thêm vào cuối dãy, còn khi lấy phần tử ra sẽ lấy ở phần tử đứng đầu dãy).
* Close: là tập hợp các đỉnh đã xét, đã duyệt qua.
* s: là đỉnh xuất phát, đỉnh gốc ban đầu trong quá trình tìm kiếm.
* g: đỉnh đích cần tìm.
* p: đỉnh đang xét, đang duyệt.

Trình bày thuật giải:

* Bước 1: Tập Open chứa đỉnh gốc s chờ được xét.
* Bước 2: Kiểm tra tập Open có rỗng không.
  + Nếu tập Open không rỗng, lấy một đỉnh ra khỏi tập Open làm đỉnh đang xét p. Nếu p là đỉnh g cần tìm, kết thúc tìm kiếm.
  + Nếu tập Open rỗng, tiến đến bước 4.
* Bước 3: Đưa đỉnh p vào tập Close, sau đó xác định các đỉnh kề với đỉnh p vừa xét. Nếu các đỉnh kề không thuộc tập Close, đưa chúng vào cuối tập Open. Quay lại bước 2.
* Bước 4: Kết luận không tìm ra đỉnh đích cần tìm.

1. Chạy ví dụ:

Input:

22,18

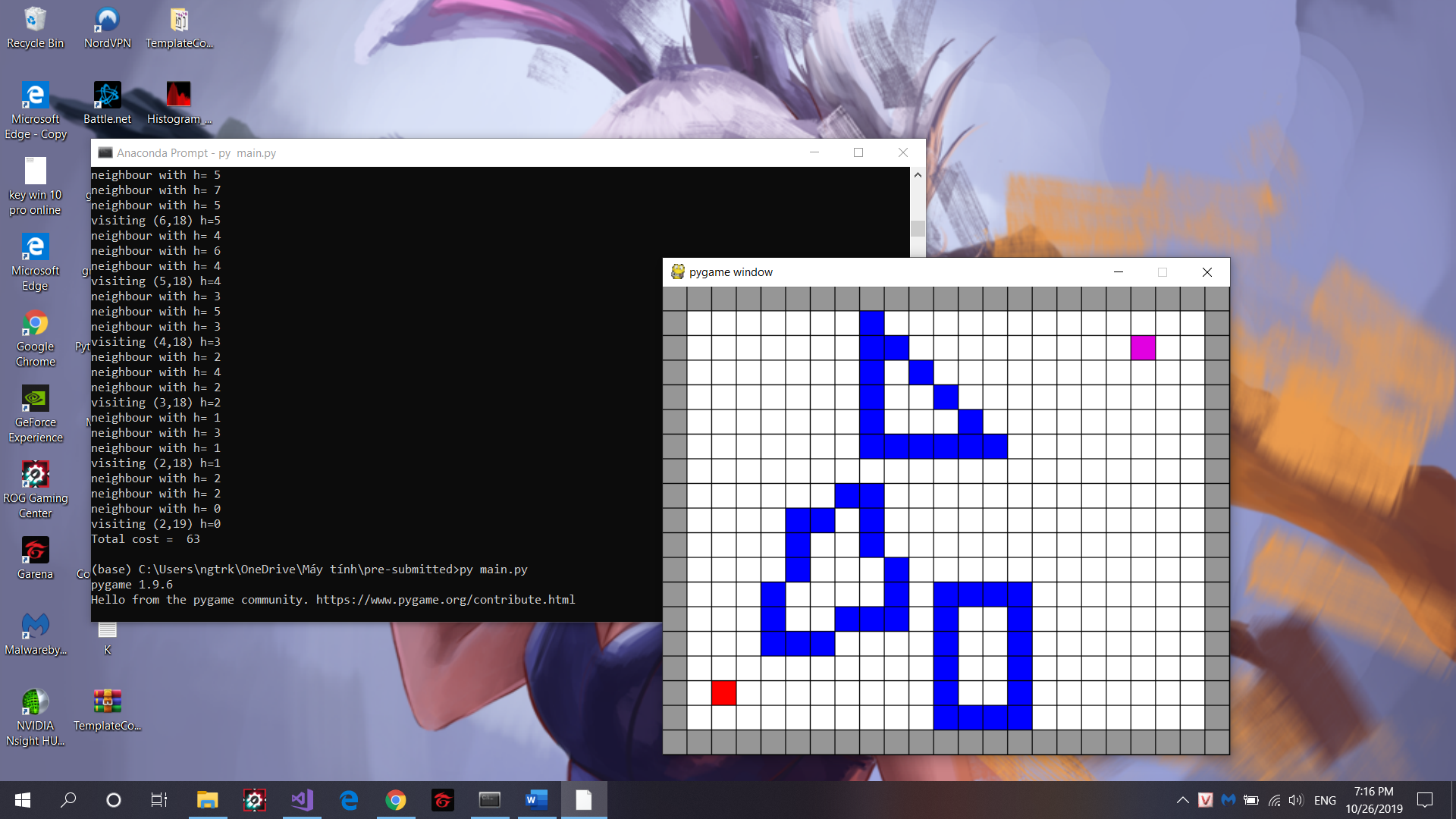
2,2,19,16

3

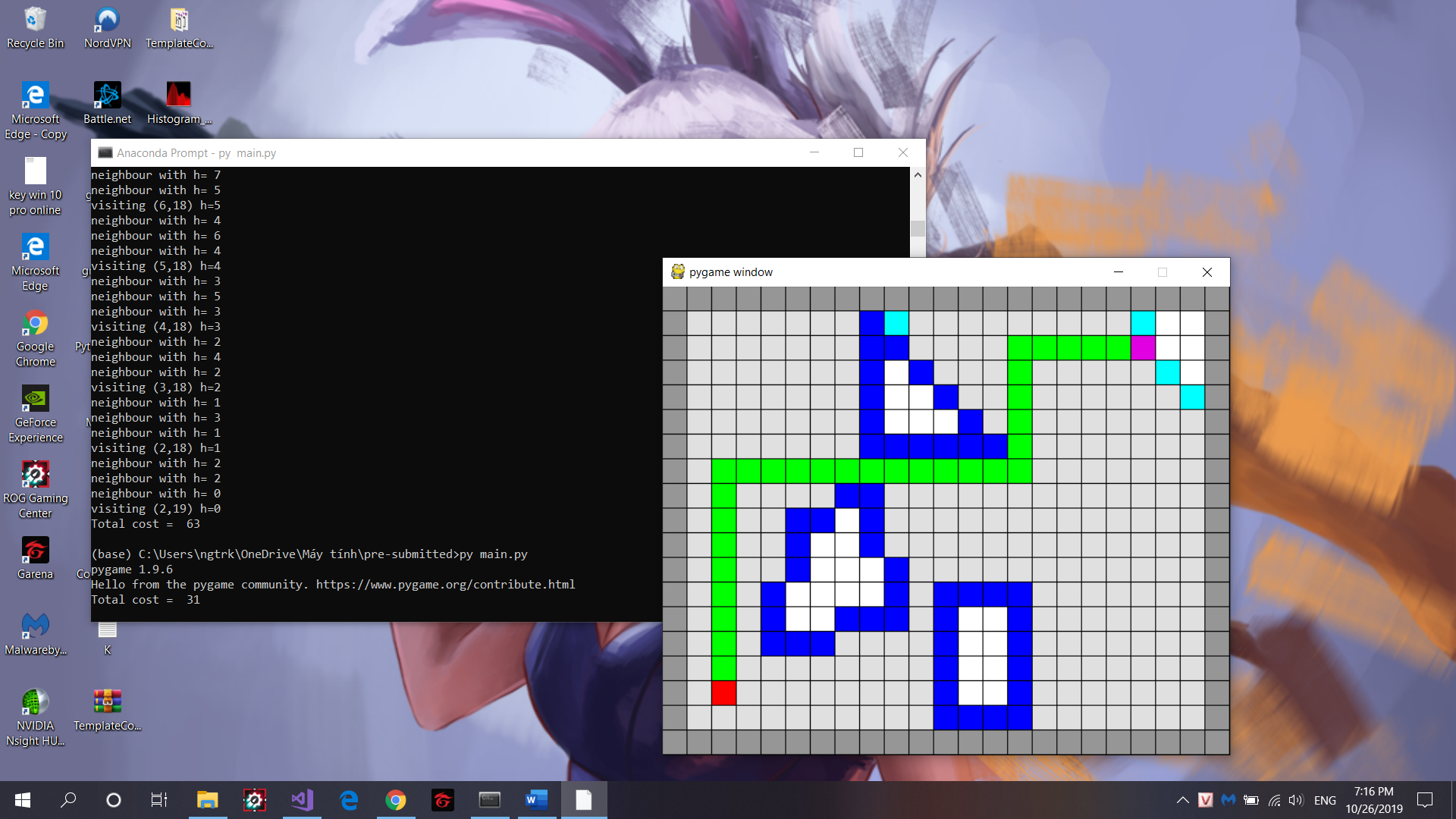
4,4,5,9,8,10,9,5

8,12,8,17,13,12

11,1,11,6,14,6,14,1

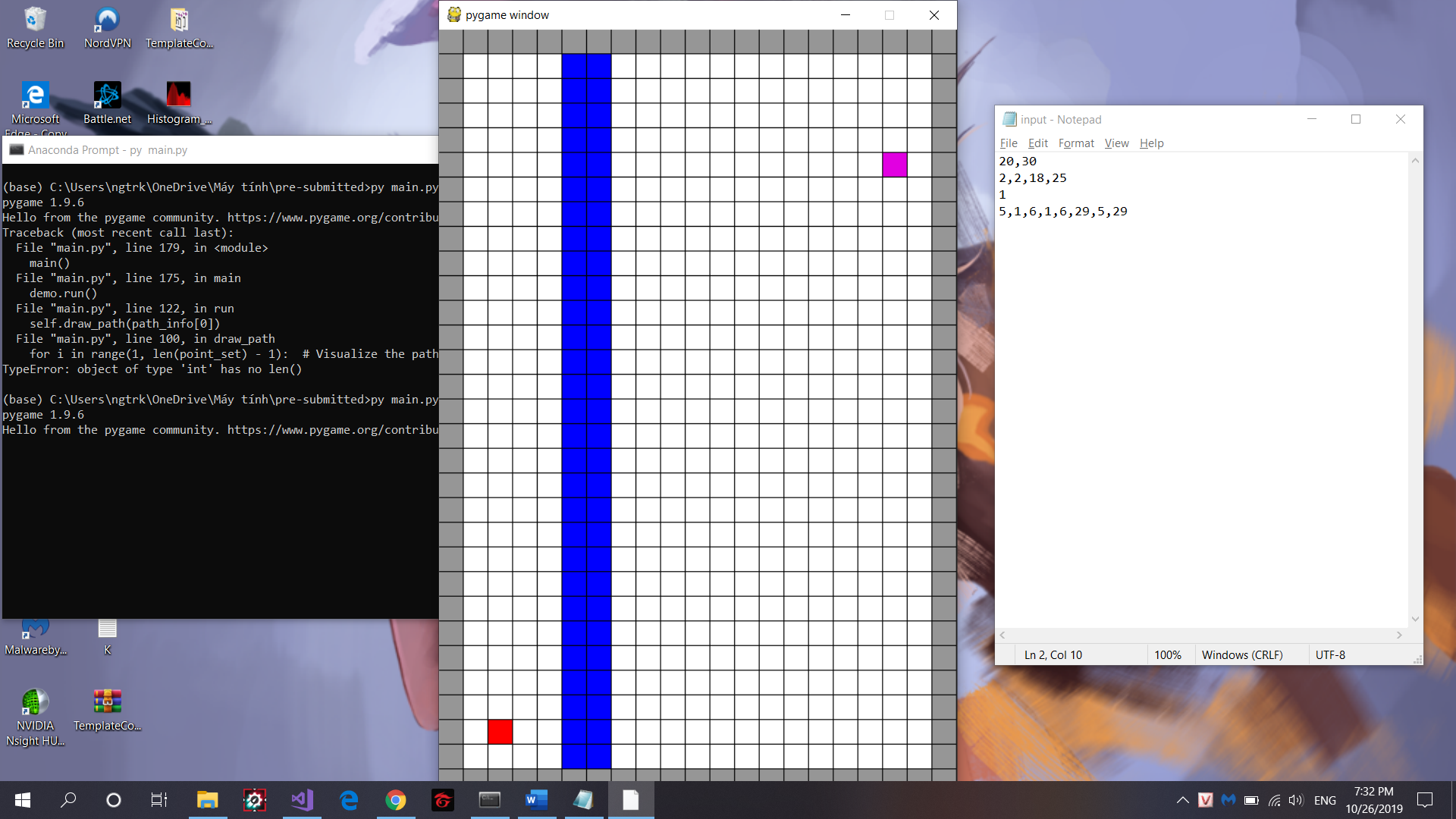


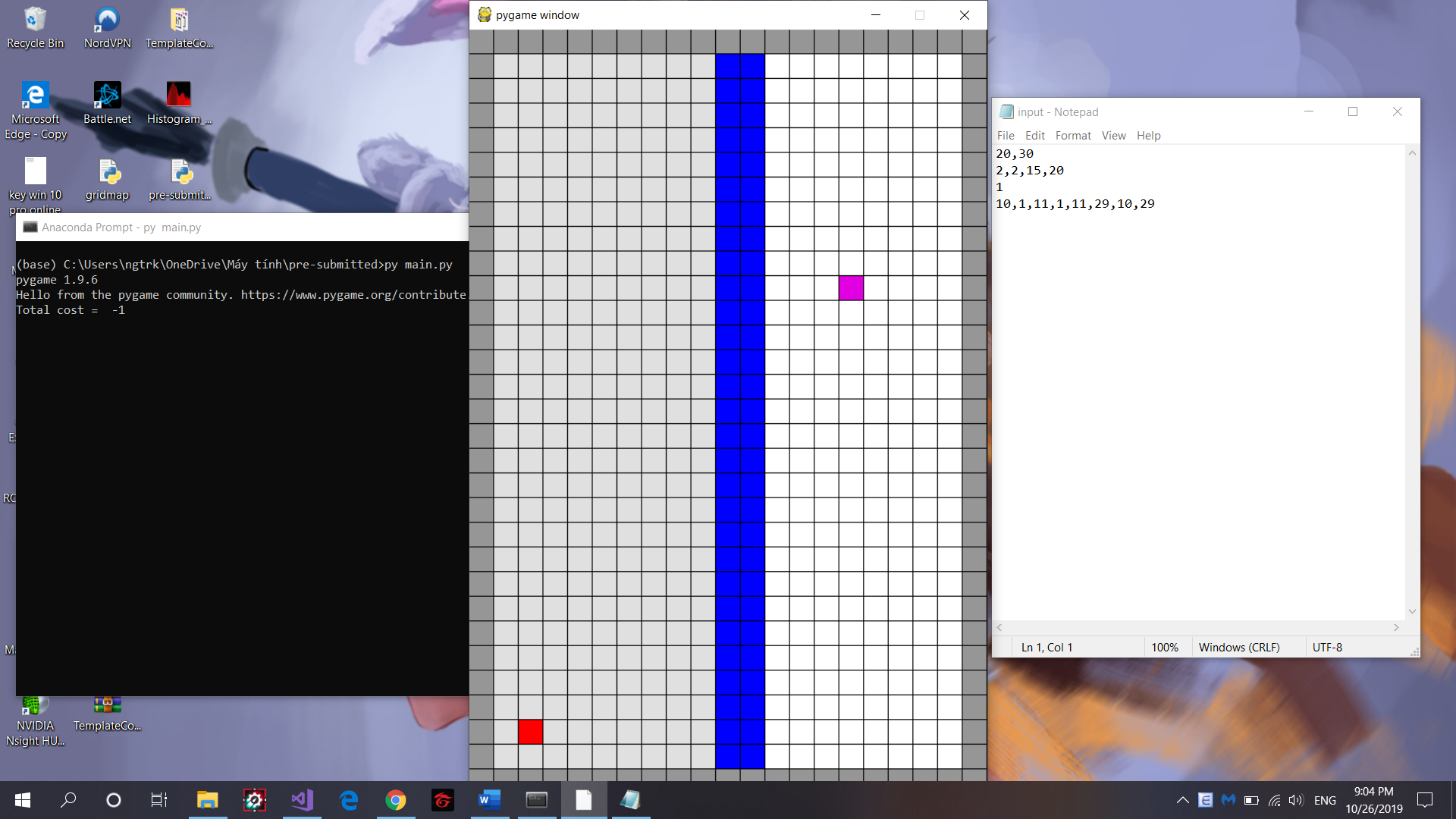
Out put:



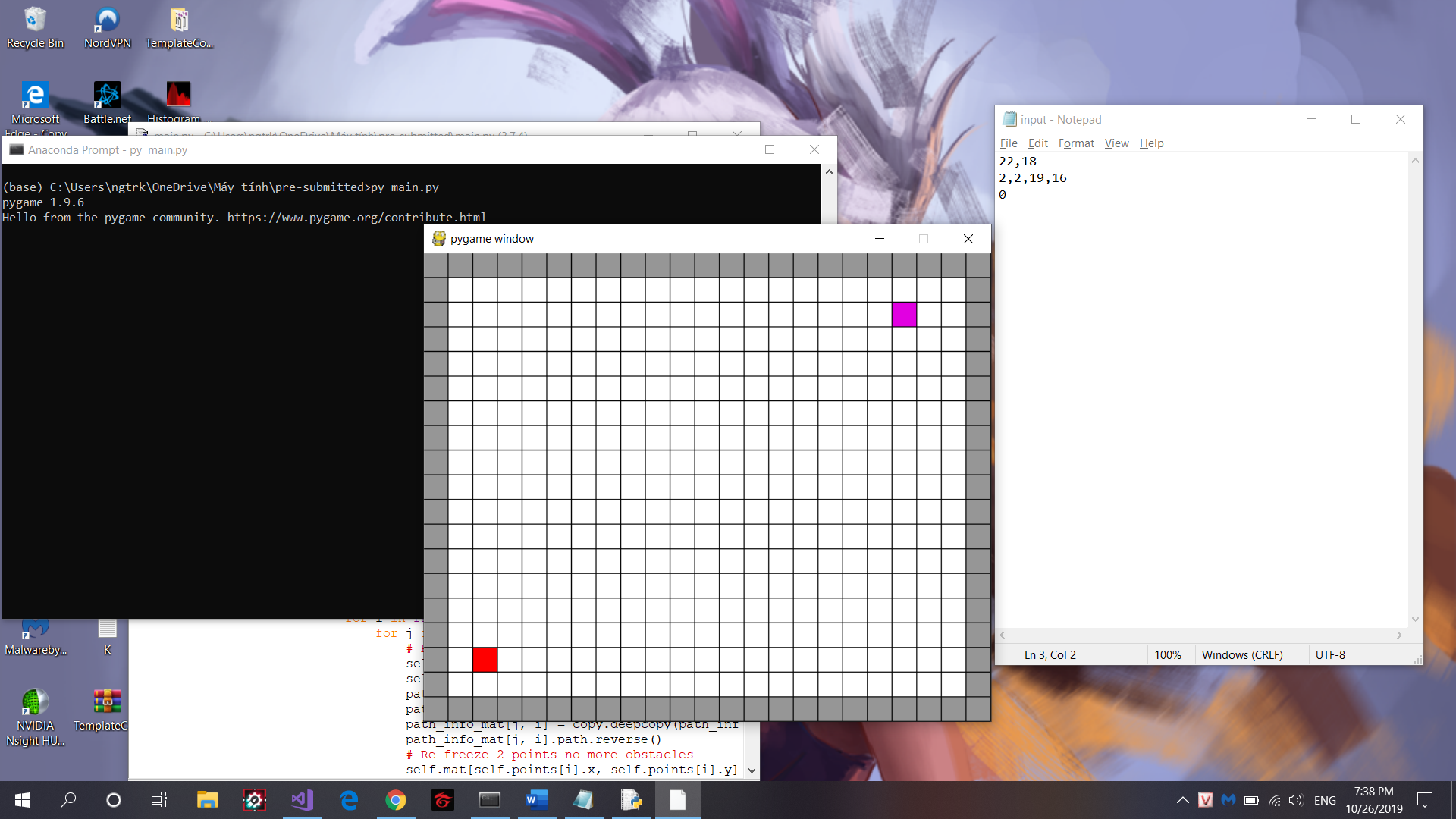
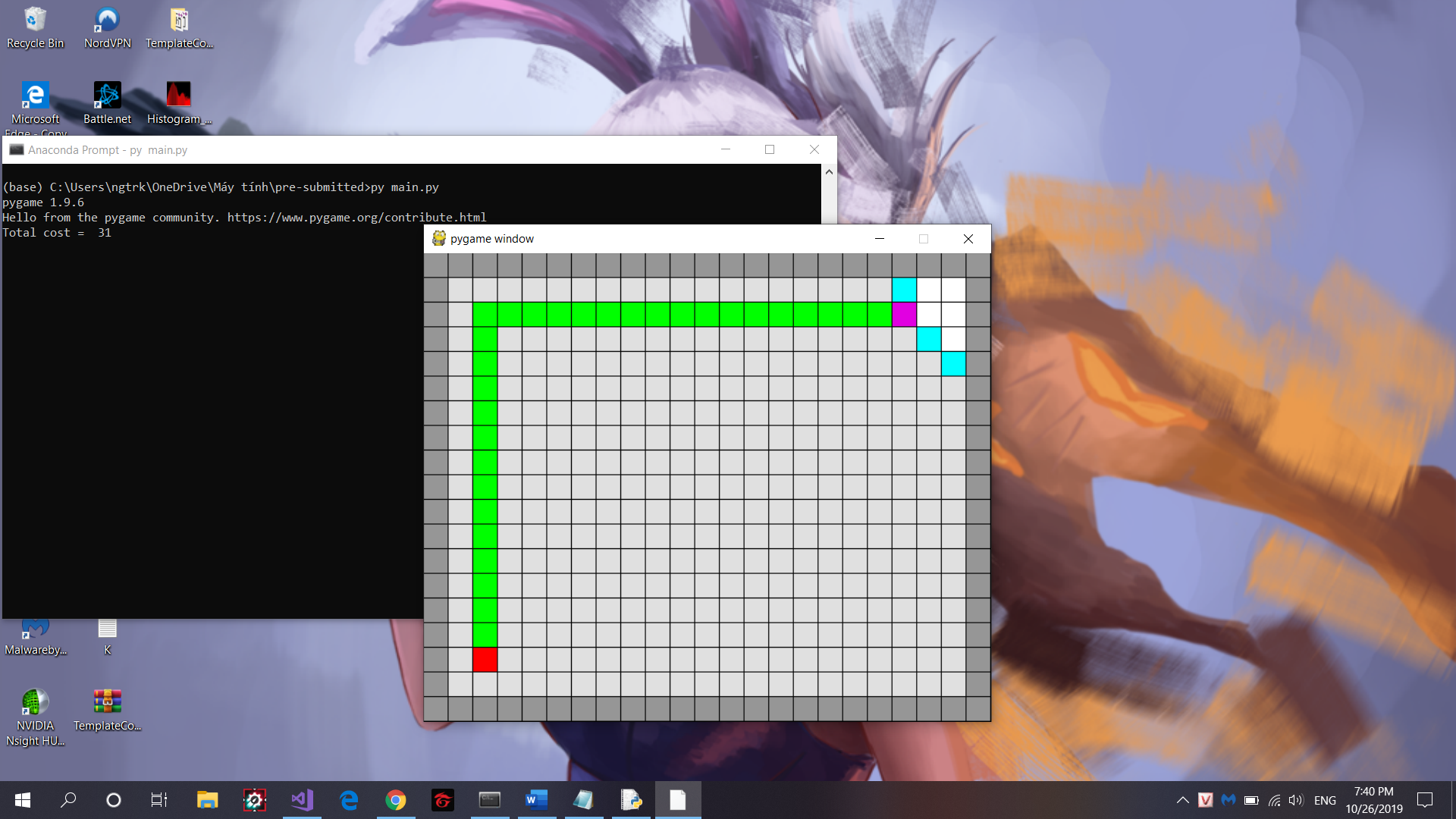
1. Trường hợp đặc biệt:

Không có đường đi





Không có vật cản:



1. Nhận xét:

* Ưu điểm:
* Xét duyệt qua tất cả đỉnh để trả về kết quả.
* Nếu số đỉnh là hữu hạn thì chắc chắn tìm ra kết quả.
* Khuyết điểm:
* Mang tính chất vét cạn, khi không gian tìm kiếm lớn không hiệu quả vì chi phí không gian và thời gian lớn.
* Mang tính mù quáng, duyệt tất cả các đỉnh, không chú ý đến thông tin trong các đỉnh để duyệt hiệu quả.

## 3.2 Greedy best-first search

1. Thuật toán:

B1. Đặt OPEN chứa trạng thái khởi đầu.

B2. Cho đến khi tìm được trạng thái đích hoặc không còn nút nào trong OPEN, thực hiện :

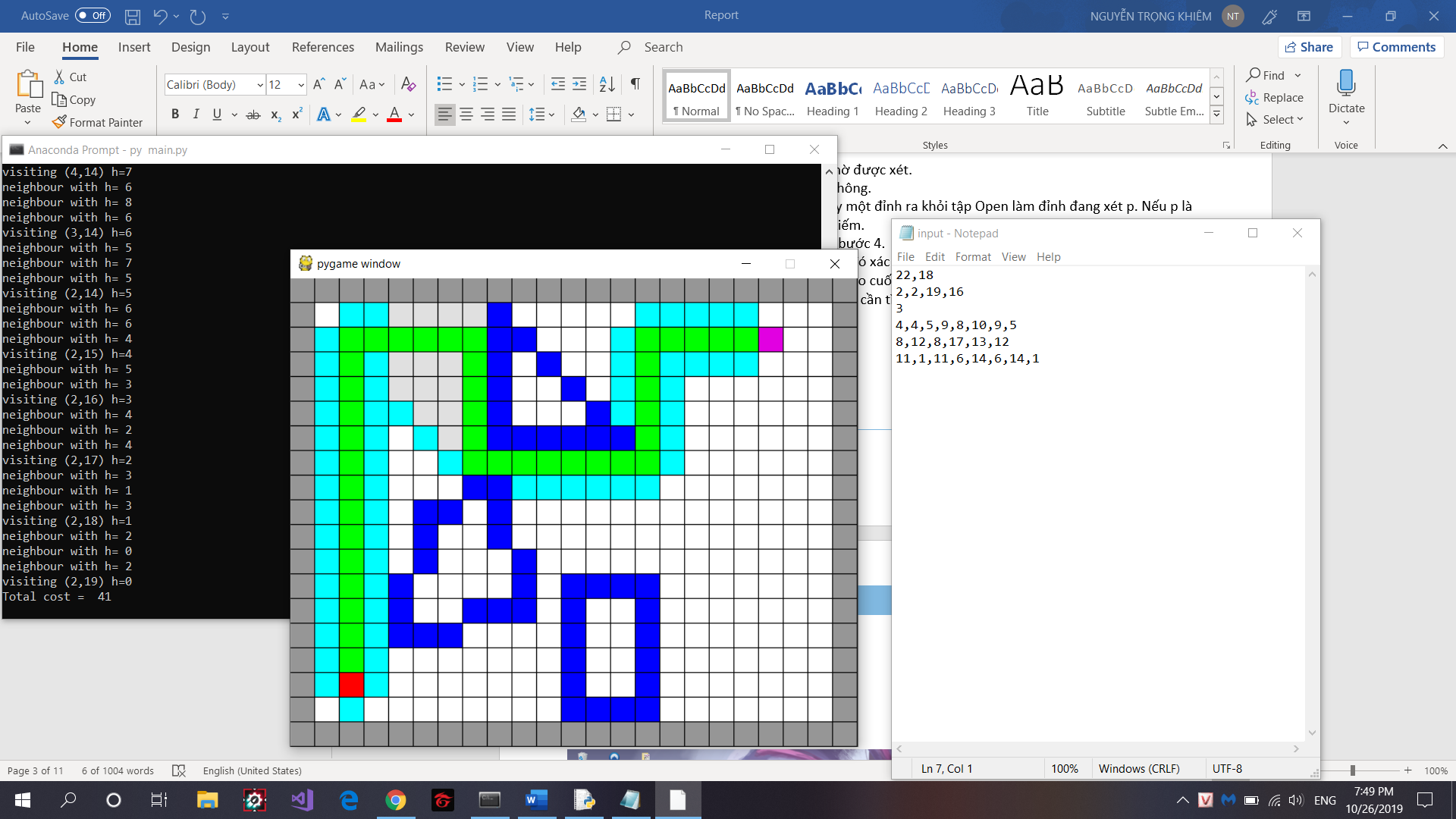
B2.a. Chọn trạng thái tốt nhất (Tmax) trong OPEN (và xóa Tmax khỏi OPEN)

B2.b. Nếu Tmax là trạng thái kết thúc thì thoát.

B2.c. Ngược lại, tạo ra các trạng thái kế tiếp Tk có thể có từ trạng thái Tmax. Đối với mỗi trạng thái kế tiếp Tk thực hiện :

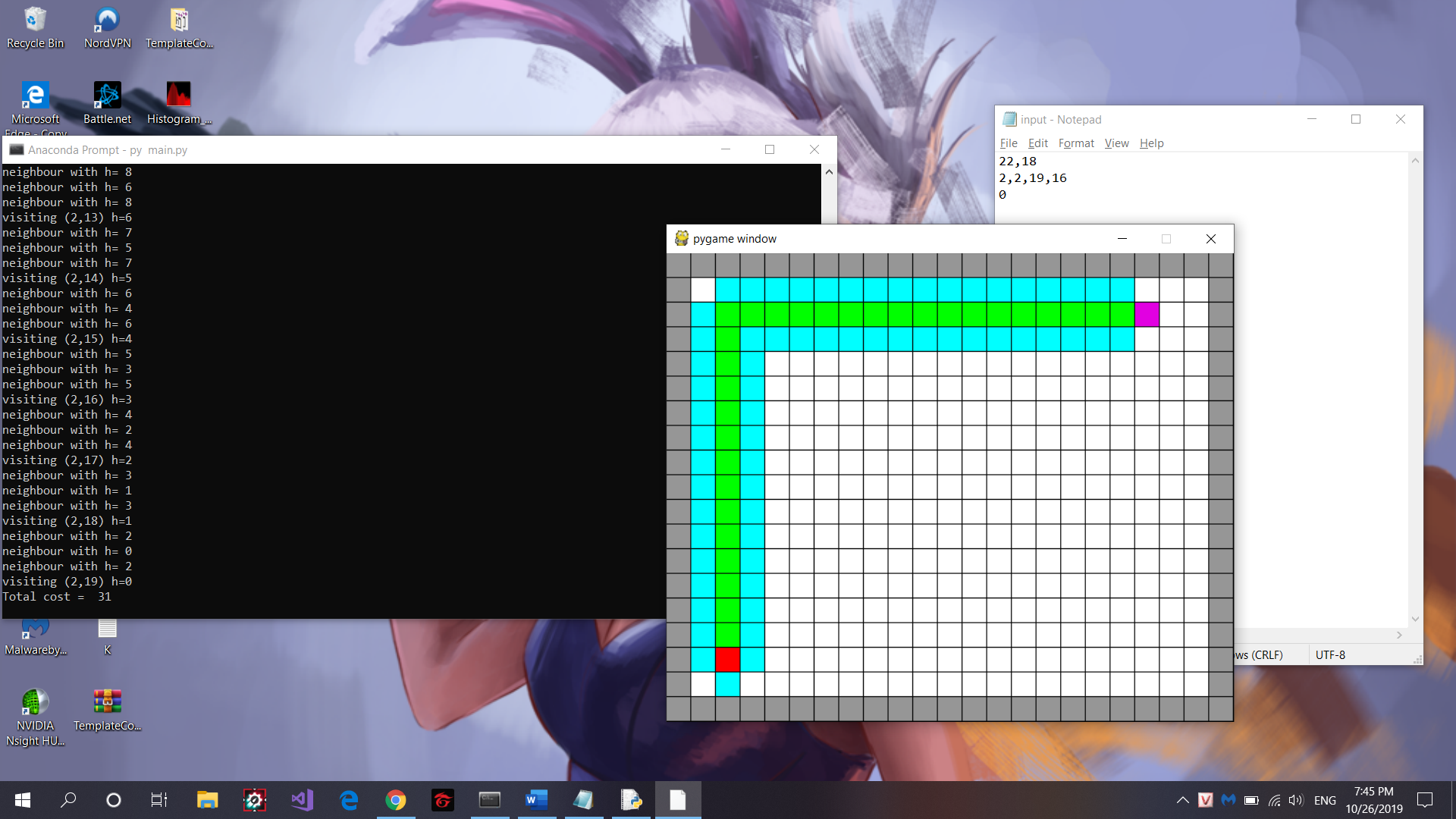
* Tính f(Tk); Thêm Tk vào OPEN

1. Chạy ví dụ:

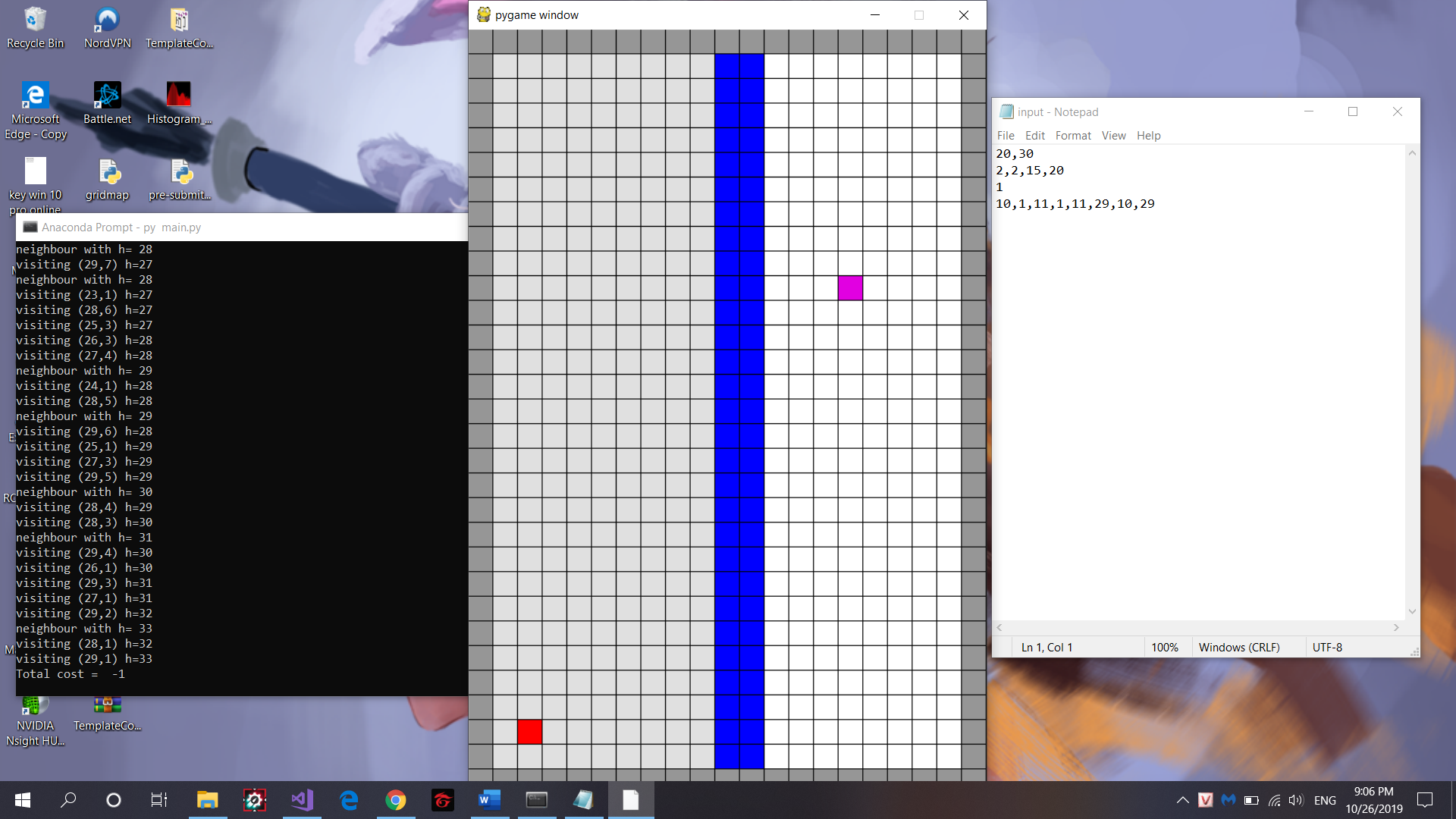


1. Trường hợp đặc biệt:

Không có vật cản



Không có đường đi



1. Nhận xét:

* Ưu điểm:
* Với hàm heuristic tốt có thể mang lại cải thiện về thời gian và không gian.
* Khuyết điểm:
* Quá trình tìm kiếm có thể đi xa khỏi lời giải. Kỹ thuật này chỉ xét một phần của không gian và coi đó là phần hứa hẹn hơn cả. Có thể bị kẹt trong vòng lặp. (Không complete).
* Chi phí lưu trữ lớn vì lưu toàn bộ các node.
* Không tối ưu.

## 3.3 A\* Search:

1. Thuật toán:

A\* lưu giữ một tập các lời giải chưa hoàn chỉnh, nghĩa là các đường đi qua đồ thị, bắt đầu từ nút xuất phát. Tập lời giải này được lưu trong một hàng đợi ưu tiên (priority queue). Thứ tự ưu tiên gán cho một đường đi *x* được quyết định bởi *f(x)=g(x)+h(x)*

Trong đó, *g(x)* là chi phí của đường đi cho đến thời điểm hiện tại, nghĩa là tổng trọng số của các cạnh đã đi qua. *h(x)* là hàm đánh giá heuristic về chi phí nhỏ nhất để đến đích từ *x*. Ví dụ, nếu "chi phí" được tính là khoảng cách đã đi qua, khoảng cách đường chim bay giữa hai điểm trên một bản đồ là một đánh giá heuristic cho khoảng cách còn phải đi tiếp.

Hàm *f(x)* có giá trị càng thấp thì độ ưu tiên của x càng cao.

Quy ước:

* Gọi Open: tập các trạng thái đã được sinh ra nhưng chưa được xét đến.
* Close: tập các trạng thái đã được xét đến.
* Cost(p, q): là khoảng cách giữa p, q.
* g(p): khoảng cách từ trạng thái đầu đến trạng thái hiện tại p.
* h(p): giá trị được lượng giá từ trạng thái hiện tại đến trạng thái đích.
* f(p) = g(p) + h(p).

Giải thuật:

* 1. Bước 1:
     + Open: = {s}
     + Close: = {}
  2. Bước 2: while (Open !={})
     + Chọn trạng thái ( đỉnh ) tốt nhất p trong Open (xóa p khỏi Open).
     + Nếu p là trạng thái kết thúc thì thoát.
     + Chuyển p qua Close và tạo ra các trạng thái kế tiếp q sau p.
       1. Nếu q đã có trong Open
          - Nếu g(q) > g(p) + Cost(p, q)

g(q) = g(p) + Cost(p, q)

f(q) = g(q) + h(q)

prev(q) = p // đỉnh cha của q là p

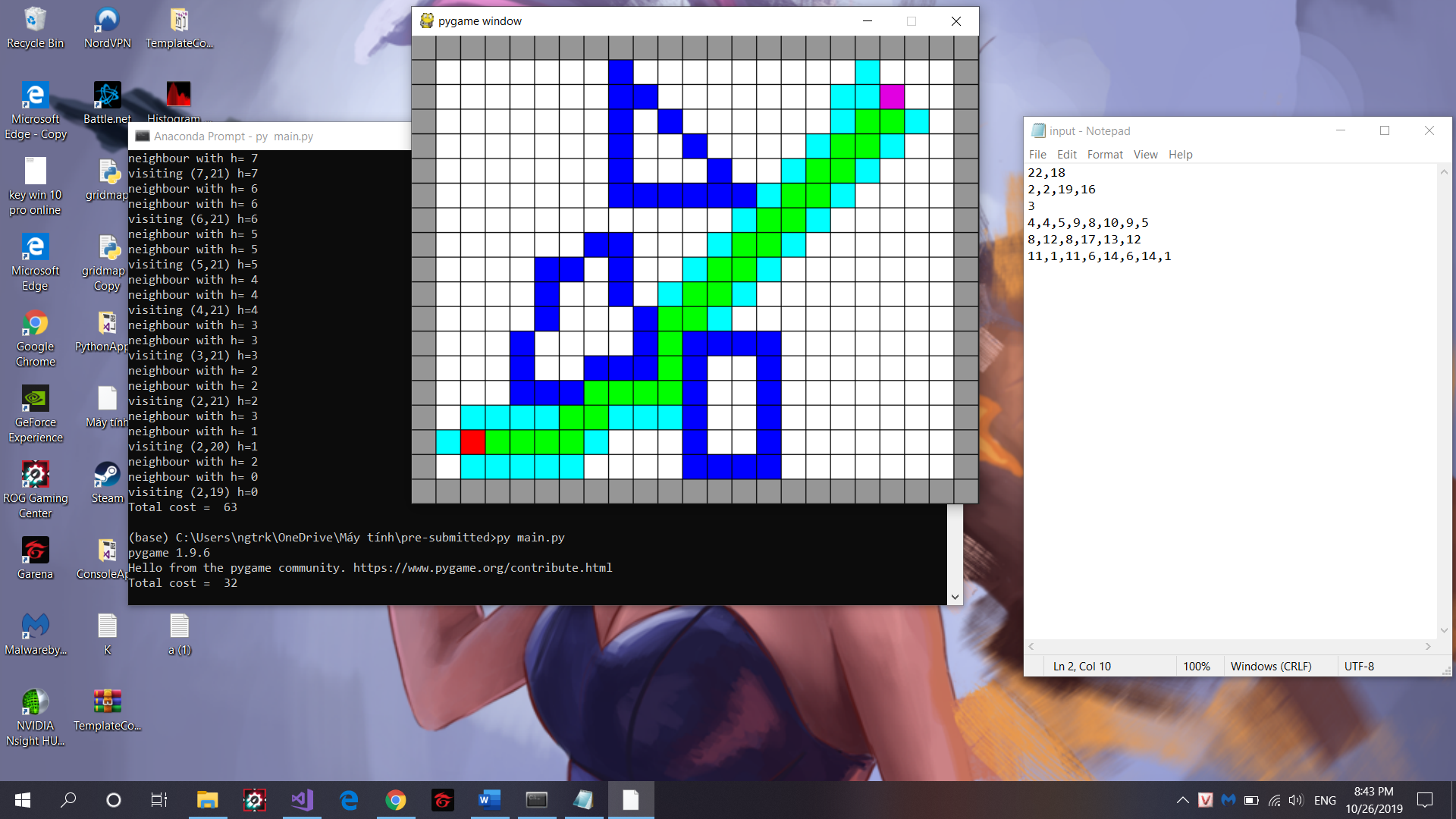
* + - 1. Nếu q chưa có trong Open
         * g(q) = g(p) + cost(p, q)
         * f(q) = g(q) + h(q)
         * prev(q) = p
         * Thêm q vào Open
      2. Nếu q có trong Close
         * Nếu g(q) > g(p) + Cost(p, q)

Bỏ q khỏi Close

Thêm q vào Open

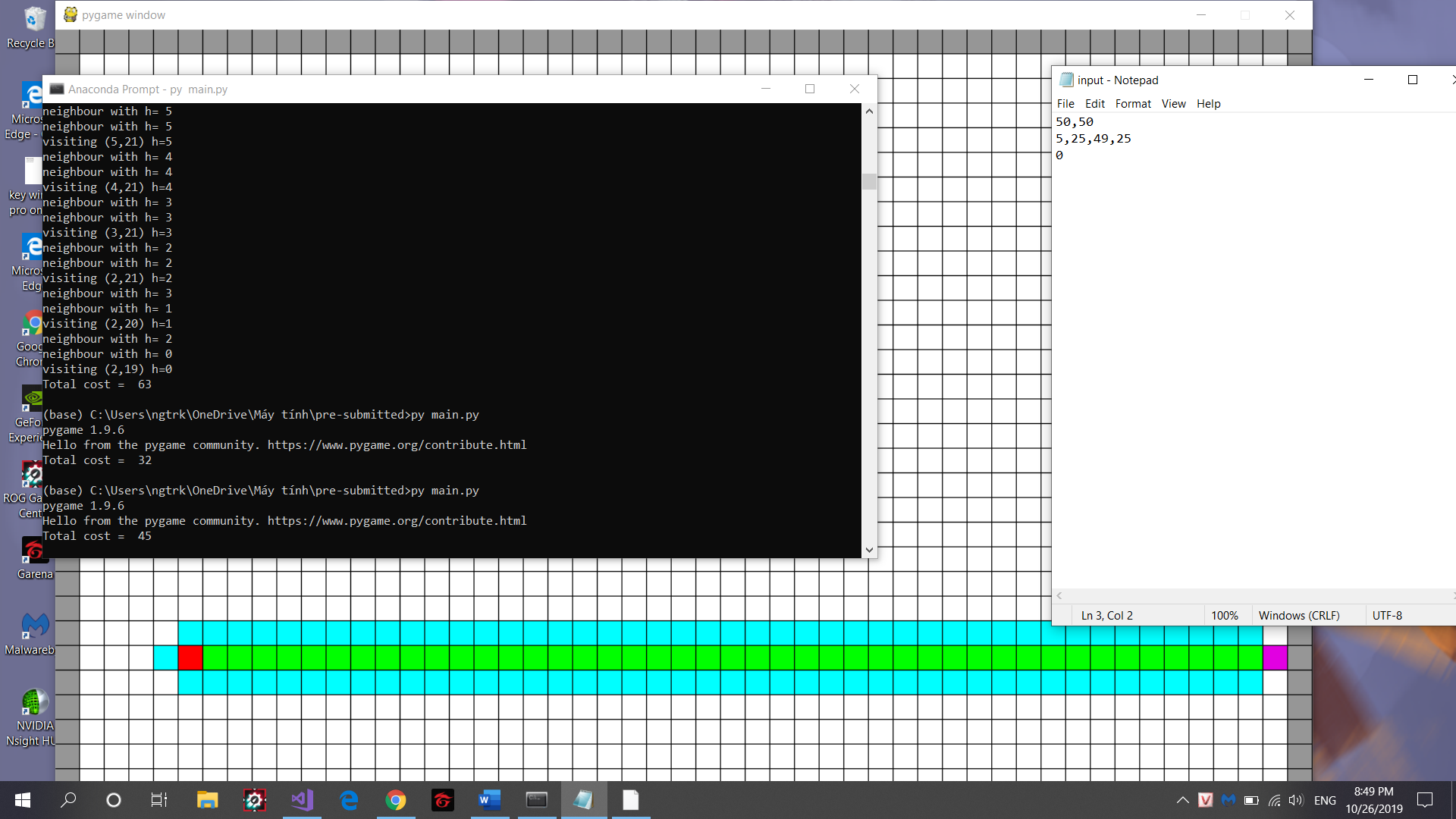
* 1. Bước 3: Không tìm được.

1. Chạy ví dụ:



1. Trường hợp đặc biệt:

Không có vật cản



Không có đường đi:



1. Nhận xét:

* Ưu điểm:
* Luôn tìm thấy một lời giải nếu bài toán có lời giải.
* A\* luôn mở rộng về phía Goal, nhưng vẫn thực hiện đánh cược để đảm bảo sự tối ưu.
* Khuyết điểm:
* Tốn bộ nhớ để lưu những trạng thái đã đi qua.
* Nếu không gian các trạng thái là hữu hạn và không có giải pháp để tránh việc xét (lặp) lại các trạng thái, thì giải thuật A\* là không hoàn chỉnh (không đảm bảo tìm được lời giải).

## 3.4 Tìm kiếm có điểm đón:

1. Ý tưởng và cách thực hiện:

Bằng các thuật toán đã cài đặt để tìm đường đi từ 1 điếm tất cả các điểm khác (điểm đầu, điểm đích và các điểm đón – trừ chính nó). Sau đó hoán vị tất các các điểm đón sau đó tính tổng đường đi với mỗi hoán vị (quy tắc: điểm đầu->{hoán vị các điểm đón}-> điểm đích). Từ đó đảm bảo tổng đường đi là nhỏ nhất.

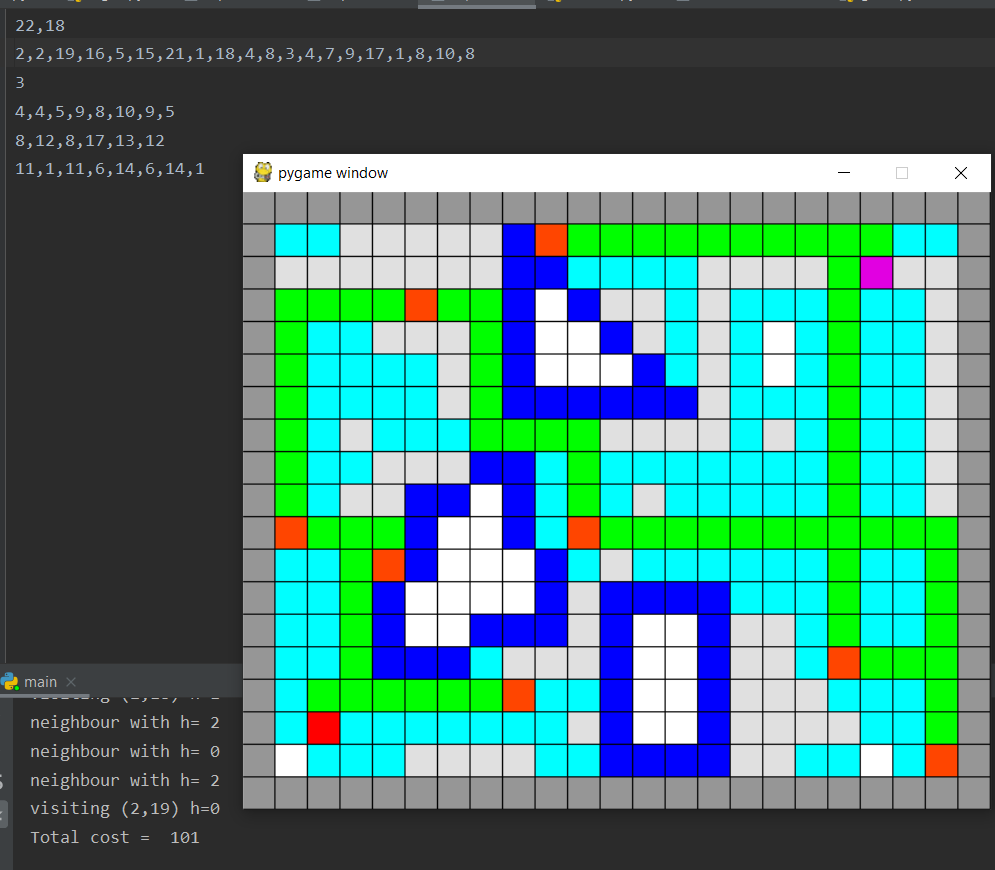
-Cách thực hiện:

* Khởi tạo ma trận chứa thông tin đường đi giữa 2 điểm trong các điểm “quan trọng” (điểm đầu, điểm đích, các điểm đón). Thứ tự chỉ số sẽ được sắp xếp theo đúng từ 0->N-1 (trong đó 0: chỉ số điểm đầu, N-1: chỉ số điểm cuối với N là số các điểm “quan trọng”)
* Nhận thấy ta phải tốn N\*N lần tính cho N điểm “quan trọng” và thông tin đường đi giữa 2 điểm A->B và B->A bất kì là như nhau chỉ đảo ngược thứ tự bước đi (tức ma trận có thể coi là “đối xứng” qua đường chéo chính). Nên ta chỉ cần tính **nửa trên** của ma trận và lấy đối xứng qua đường chéo chính (kèm đảo thứ tự bước đi).
* Với mỗi chuỗi hoán vị từ tập chỉ số của điểm đón {1,2,…,n-2} ta sẽ tính tổng đường đi (từ đường đi của 2 điểm lấy trong ma trận). Đồng thời tìm min của các tổng đường đi kèm chuỗi hoán vị tương ứng.

1. Nhận xét:

* Ưu điểm: Ý tưởng đơn giản, dễ hiểu. Nếu đường đi giữa các điểm là nhỏ nhất thì yêu cầu tổng đường đi là nhỏ nhất luôn được đảm bảo.
* Nhược điểm: Tốn chi phí cao. Chi phí hoán vị để tính tổng đường đi là K! với K là số các điểm đón. Kèm theo tính tổng đường đi với mỗi hoán vị thì chi phí sẽ là O(K! \*N). Đó là vẫn chưa tính đến chi phí tính đường đi giữa 2 điểm trong tập điểm “quan trọng” để bỏ vào trong ma trận.

1. Chạy ví dụ:



# Các lựu chọn khi cài đặt

Các thuật toán cài đặt chỉ cho phép đi dọc và ngang với chi phí mỗi bước là 1.

Riêng phần đi có điểm đón cho phép đi lên đường đi đã đi trước đó. Vì nếu có điểm đón nằm ở vi trí chỉ có 1 đường đi vào thì đường đi ra phải trùng với đường đi vào.

